GROUP 5
Comparative Analysis for Diabetic Retinopathy Detection: A case study on Debrecen Dataset

Praneeth Puppala	**2315346**
Saran Teja Mallela	**2315340**
Tanmai Veerapaneni	**2315355**
Eeraboina Keerthi Yadav	**2311912**
Kunduru Neha	**2351359**

Abstract

Diabetic Retinopathy (DR) is a leading cause of blindness globally, emphasizing the need for accurate and efficient diagnostic tools. The analysis involves Data preprocessing, feature selection, hyperparameter tuning and testing of several models. This study presents a comparative analysis of machine learning models for DR detection using the Debrecen dataset. **Logistic Regression, Support Vector Machine (SVM), Random Forest, and K-Nearest Neighbors (KNN)** models were implemented and evaluated. Preprocessing techniques including **winsorization, KNN Imputation, feature scaling** and dimensionality reduction were applied to enhance model performance. Performance metrics such as accuracy, precision, recall, f1-score and support were utilized for evaluation. Model performance evaluation via accuracy, precision, recall, and F1-score reveals strengths and weaknesses of each model. Results indicate that **SVM outperforms Regression and all other machine learning algorithms**, achieving the highest accuracy. However, Regression, RandomForest, and KNN exhibit competitive performance, underscoring their effectiveness in DR detection. This comparative analysis provides insights into the strengths and limitations of various machine learning approaches for DR diagnosis, facilitating informed decision-making in clinical settings.

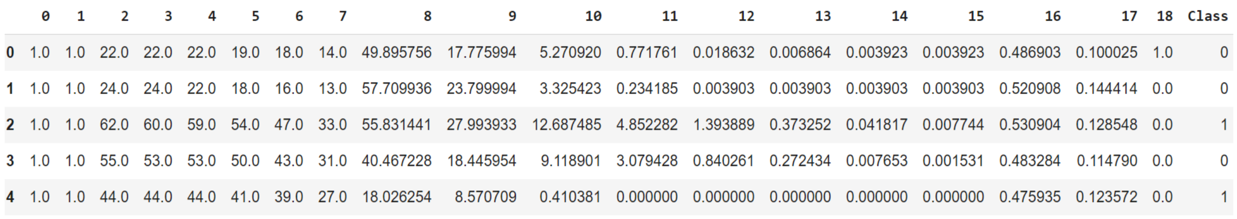
1. Introduction

Diabetic Retinopathy (DR) is a progressive eye disease affecting a significant portion of the diabetic population worldwide. It is characterized by damage to the blood vessels of the retina due to prolonged exposure to high blood sugar levels. DR, if left untreated, can lead to vision impairment and even blindness. Early detection and timely intervention are crucial in preventing irreversible vision loss.
In recent years, the advancement of machine learning techniques has shown promise in improving the accuracy and efficiency of DR detection. Leveraging these techniques, researchers have developed various algorithms and models capable of analyzing retinal images to identify signs of DR. One such dataset widely used for research purposes is the Debrecen Diabetic Retinopathy Dataset.
The Debrecen dataset consists of retinal images acquired using fundus photography, along with corresponding labels indicating the presence or absence of diabetic retinopathy. These images serve as valuable resources for training and evaluating machine learning models for DR detection.
This project aims to explore and compare the performance of different machine learning algorithms in detecting diabetic retinopathy using the Debrecen dataset. Specifically, Support Vector Machine (SVM), Random Forest, K-Nearest Neighbors (KNN), Naive Bayes, and Convolutional Neural Network (CNN) models will be implemented and evaluated.

Through this comparative analysis, insights into the strengths and weaknesses of each model will be gained, facilitating informed decision-making regarding the choice of algorithm for DR detection. Additionally, the project seeks to contribute to the ongoing efforts to enhance the accuracy and accessibility of diabetic retinopathy screening, ultimately improving patient outcomes and reducing the burden of vision loss associated with this debilitating condition.

2. Data Description

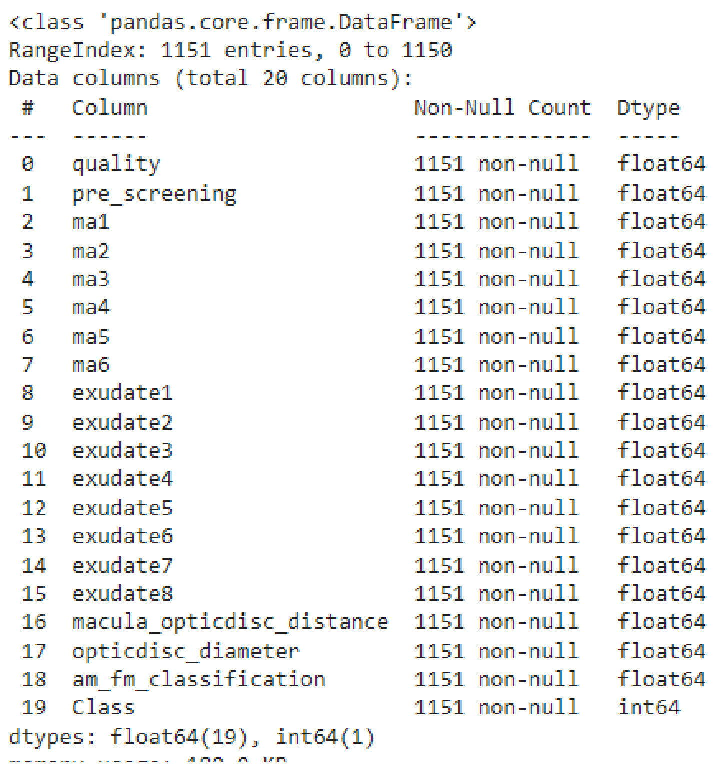
The diabetic retinopathy detection dataset used in this project comprises 1151 instances, each associated with 19 features. These features include a combination of integer and real values, reflecting various attributes extracted from retinal images through fundus photography. An example csv description is shown in fig 1.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	Class
0	1.0	1.0	22.0	22.0	22.0	19.0	18.0	14.0	49.895756	17.775994	5.270920	0.771761	0.018632	0.006864	0.003923	0.003923	0.486903	0.100025	1.0	0
1	1.0	1.0	24.0	24.0	22.0	18.0	16.0	13.0	57.709936	23.799994	3.325423	0.234185	0.003903	0.003903	0.003903	0.003903	0.520908	0.144414	0.0	0
2	1.0	1.0	62.0	60.0	59.0	54.0	47.0	33.0	55.831441	27.993933	12.687485	4.852282	1.393889	0.373252	0.041817	0.007744	0.530904	0.128548	0.0	1
3	1.0	1.0	55.0	53.0	53.0	50.0	43.0	31.0	40.467228	18.445954	9.118901	3.079428	0.840261	0.272434	0.007653	0.001531	0.483284	0.114790	0.0	0
4	1.0	1.0	44.0	44.0	44.0	41.0	39.0	27.0	18.026254	8.570709	0.410381	0.000000	0.000000	0.000000	0.000000	0.000000	0.475935	0.123572	0.0	1

Fig 1: Data Description [df.head() output]

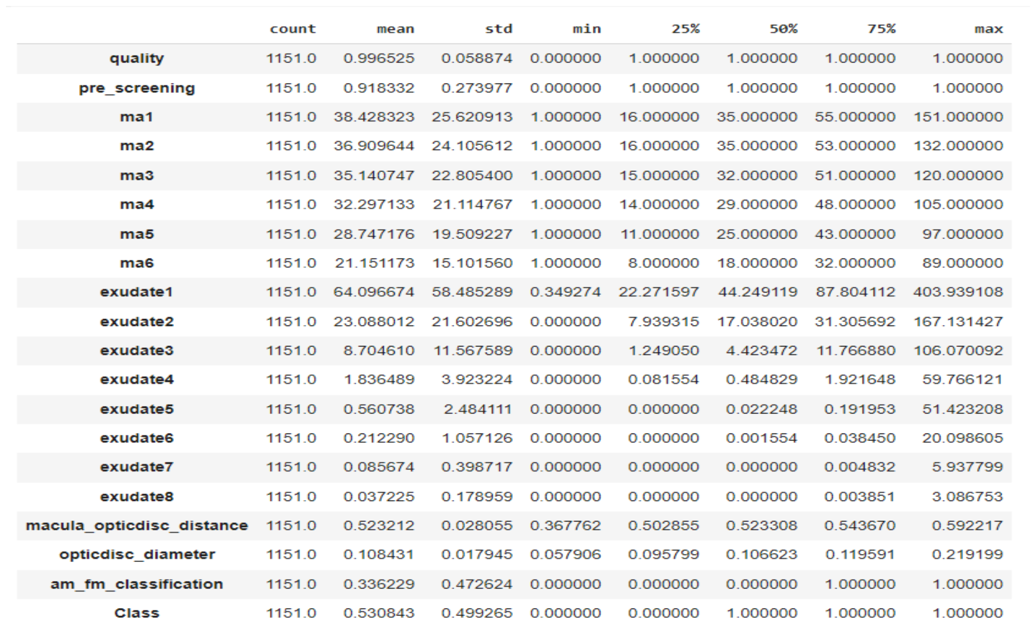
The dataset is characterized as multivariate, indicating that it contains multiple variables or features for each instance. As the subject area pertains to health and medicine, the dataset is specifically tailored for tasks related to diabetic retinopathy classification.
Each instance in the dataset corresponds to a retinal image, and the associated features provide quantitative information about different aspects of the retina, such as lesions, hemorrhages, exudates, and other pathological indicators of diabetic retinopathy.
The primary task associated with this dataset is classification, wherein machine learning models are trained to categorize retinal images into binary classes, typically indicating the presence or absence of diabetic retinopathy.



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1151 entries, 0 to 1150
Data columns (total 20 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   quality                   1151 non-null   float64
 1   pre_screening             1151 non-null   float64
 2   ma1                       1151 non-null   float64
 3   ma2                       1151 non-null   float64
 4   ma3                       1151 non-null   float64
 5   ma4                       1151 non-null   float64
 6   ma5                       1151 non-null   float64
 7   ma6                       1151 non-null   float64
 8   exudate1                  1151 non-null   float64
 9   exudate2                  1151 non-null   float64
 10  exudate3                  1151 non-null   float64
 11  exudate4                  1151 non-null   float64
 12  exudate5                  1151 non-null   float64
 13  exudate6                  1151 non-null   float64
 14  exudate7                  1151 non-null   float64
 15  exudate8                  1151 non-null   float64
 16  macula_opticdisc_distance 1151 non-null   float64
 17  opticdisc_diameter        1151 non-null   float64
 18  am_fm_classification      1151 non-null   float64
 19  Class                     1151 non-null   int64
dtypes: float64(19), int64(1)
```

Fig 2: Data types and information of each column

Given the diverse nature of the features, including both integer and real values, the dataset offers a comprehensive representation of retinal characteristics relevant to diabetic retinopathy detection. This diversity enables the development and evaluation of robust machine learning models capable of accurately identifying signs of diabetic retinopathy in retinal images.



	count	mean	std	min	25%	50%	75%	max
quality	1151.0	0.996525	0.058874	0.000000	1.000000	1.000000	1.000000	1.000000
pre_screening	1151.0	0.918332	0.273977	0.000000	1.000000	1.000000	1.000000	1.000000
ma1	1151.0	38.428323	25.620913	1.000000	16.000000	35.000000	55.000000	151.000000
ma2	1151.0	36.909644	24.105612	1.000000	16.000000	35.000000	53.000000	132.000000
ma3	1151.0	35.140747	22.805400	1.000000	15.000000	32.000000	51.000000	120.000000
ma4	1151.0	32.297133	21.114767	1.000000	14.000000	29.000000	48.000000	105.000000
ma5	1151.0	28.747176	19.509227	1.000000	11.000000	25.000000	43.000000	97.000000
ma6	1151.0	21.151173	15.101560	1.000000	8.000000	18.000000	32.000000	89.000000
exudate1	1151.0	64.096674	58.485289	0.349274	22.271597	44.249119	87.804112	403.939108
exudate2	1151.0	23.088012	21.602696	0.000000	7.939315	17.038020	31.305692	167.131427
exudate3	1151.0	8.704610	11.567589	0.000000	1.249050	4.423472	11.766880	106.070092
exudate4	1151.0	1.836489	3.923224	0.000000	0.081554	0.484829	1.921648	59.766121
exudate5	1151.0	0.560738	2.484111	0.000000	0.000000	0.022248	0.191953	51.423208
exudate6	1151.0	0.212290	1.057126	0.000000	0.000000	0.001554	0.038450	20.098605
exudate7	1151.0	0.085674	0.398717	0.000000	0.000000	0.000000	0.004832	5.937799
exudate8	1151.0	0.037225	0.178959	0.000000	0.000000	0.000000	0.003851	3.086753
macula_opticdisc_distance	1151.0	0.523212	0.028055	0.367762	0.502855	0.523308	0.543670	0.592217
opticdisc_diameter	1151.0	0.108431	0.017945	0.057906	0.095799	0.106623	0.119591	0.219199
am_fm_classification	1151.0	0.336229	0.472624	0.000000	0.000000	0.000000	1.000000	1.000000
Class	1151.0	0.530843	0.499265	0.000000	0.000000	1.000000	1.000000	1.000000

Fig 3: data description [df.describe() output]

3. Data Collection and Processing

3.1. Data Collection

Download the dataset from the UCI Machine Learning Repository. The link is "https://archive.ics.uci.edu/dataset/329/diabetic+retinopathy+debrecen" . The Description of the Dataset is provided in the Section 2 i.e. Data Description. Refer Section 2 for Dataset details.

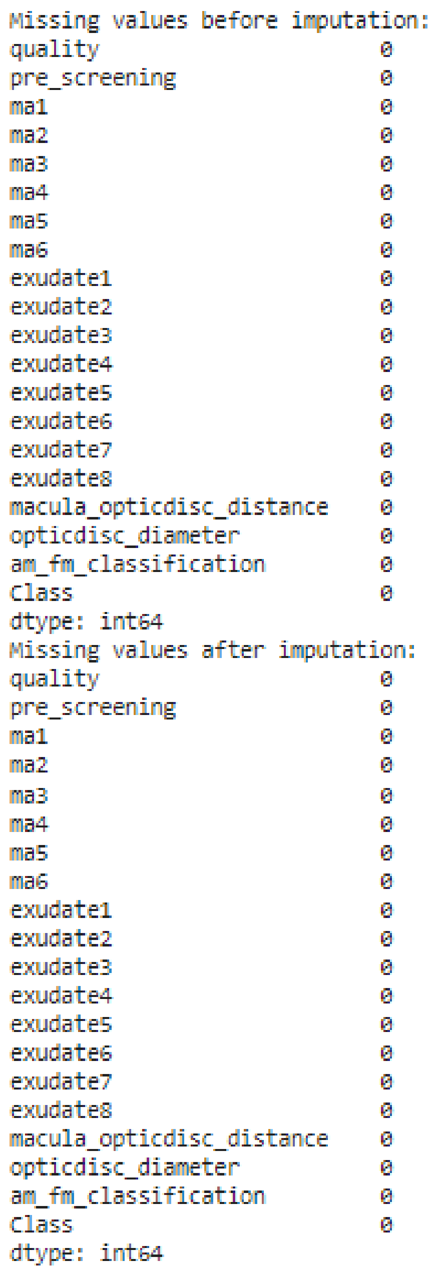
3.2. Data Preprocessing

Load the dataset into Python, **convert the arff file to csv file and conduct exploratory data analysis (EDA)** to understand its structure, features, distributions, and identify any missing values. Then preprocess the data by scaling features if necessary, and encoding categorical variables. Use Python libraries like pandas to read the dataset file (e.g., CSV format) into a DataFrame, which is a tabular data structure.

Handling Missing Values: Our provided dataset exhibited a commendable absence of missing values. This not only streamlined the pre-processing phase but also obviated the need for imputation techniques or the removal of incomplete records. The dataset's completeness across all variables ensures its reliability for analysis. **K-Nearest Neighbors (KNN) imputation** is a technique used to fill in missing values in a dataset by using the values of its nearest neighbors. See Fig 4.

Performing KNN imputation specifically for the Diabetic Retinopathy Detection on the Debrecen dataset involves using KNN to fill in missing values in the dataset attributes. Here's a general approach you can take:

1. Preprocess the data: Ensure that the dataset is properly loaded and preprocess it as necessary. This may involve handling missing values, scaling features, and splitting the data into training and testing sets.
2. Identify missing values: Determine which attributes in the dataset have missing values that need to be imputed.
3. Select a distance metric: Choose a suitable distance metric for measuring similarity between instances. Since the Debrecen dataset likely contains numerical attributes, commonly used metrics like Euclidean distance can be appropriate.
4. Choose a value of k: Determine the value of k for the KNN algorithm. This can be chosen through cross-validation or other validation techniques to find the optimal value for the dataset.
5. Implement KNN imputation: Use a KNN algorithm to impute missing values. For each missing value, find the k nearest neighbors based on the chosen distance metric and impute the missing value with the average (for numerical attributes) or mode (for categorical attributes) of these neighbors.
6. Evaluate the imputed data: After imputing missing values, evaluate the quality of the imputed dataset. This can involve checking statistical summaries, comparing distributions before and after imputation, and assessing the impact on downstream tasks like diabetic retinopathy detection.
7. Validate the results: Finally, validate the effectiveness of KNN imputation for the specific task of diabetic retinopathy detection on the Debrecen dataset. This may involve training and testing machine learning models with and without imputation and comparing their performance.



```
Missing values before imputation:
quality                      0
pre_screening                0
ma1                          0
ma2                          0
ma3                          0
ma4                          0
ma5                          0
ma6                          0
exudate1                     0
exudate2                     0
exudate3                     0
exudate4                     0
exudate5                     0
exudate6                     0
exudate7                     0
exudate8                     0
macula_opticdisc_distance    0
opticdisc_diameter           0
am_fm_classification         0
Class                        0
dtype: int64
Missing values after imputation:
quality                      0
pre_screening                0
ma1                          0
ma2                          0
ma3                          0
ma4                          0
ma5                          0
ma6                          0
exudate1                     0
exudate2                     0
exudate3                     0
exudate4                     0
exudate5                     0
exudate6                     0
exudate7                     0
exudate8                     0
macula_opticdisc_distance    0
opticdisc_diameter           0
am_fm_classification         0
Class                        0
dtype: int64
```

Fig 4: Missing data before and after imputation

Handling Outliers: Outliers are only present in sales data. These are similar to outliers belonging to one category. These cannot be removed as they represent important findings but not errors. Handling Outliers in the Diabetic Retinopathy Debrecen dataset can be crucial for building accurate predictive models. **Winsorization** replaces extreme values (outliers) with less extreme values at the tails of the distribution. This can help mitigate the influence of outliers on the model while preserving the overall distribution of the data. Refer Fig 5.

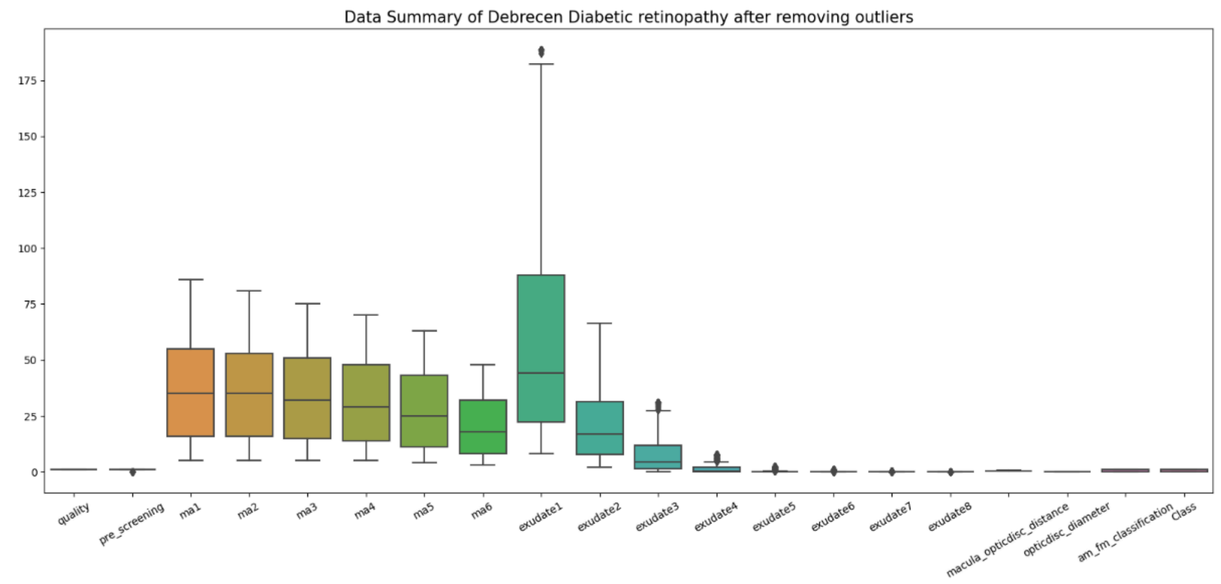


Fig 5: Data Summary after removal of outliers using winsorization

3.3. Sample Data Analysis

Explore the dataset to understand its features, data types, summary statistics, and distributions. Identify any missing values and visualize the distributions of numeric features (**Exploratory Data Analysis**). Some machine learning algorithms require feature scaling to ensure all features contribute equally to the model training process. Scale numeric features if necessary. If the dataset contains categorical variables, we'll encode them into numerical values to make them suitable for machine learning algorithms. Figure 6 shows the sample EDA.

```
Number of Normal Eye:  540
Number of DR Eye:  611
```

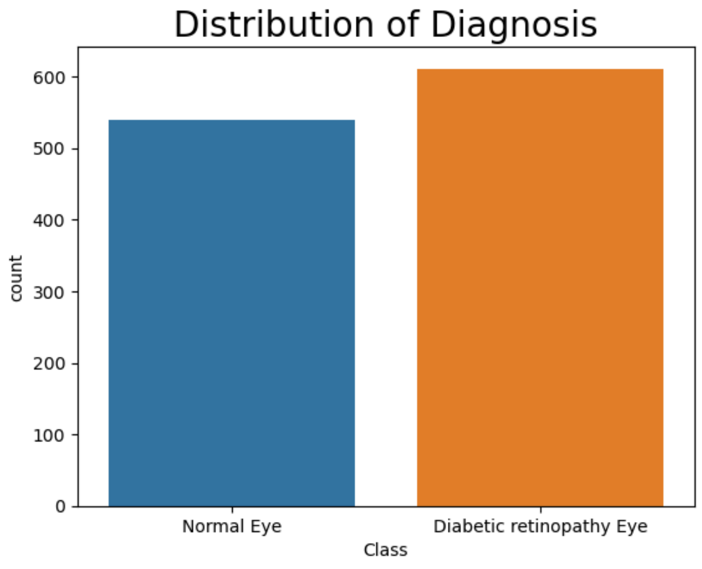


Fig 6: Distribution of Diagnosis i.e. Number of Normal eyes vs DR eyes

3.3. Feature Engineering

3.3.1. Scaling:
Scaling is a preprocessing technique used to standardize the range of numeric features in the dataset. It ensures that all features contribute equally to the model training process, preventing features with larger scales from dominating the learning algorithm. Common scaling methods include Standard Scaling (Standardization) and Min-Max Scaling (Normalization). **Standard Scaling (Z-score normalization) transforms features to have a mean of 0 and a standard deviation of 1**. Min-Max Scaling transforms features to a fixed range, usually between 0 and 1.

3.3.2. Feature Selection:
Feature selection is the process of selecting a subset of relevant features from the original set of features to improve model performance and reduce computational complexity. It helps in improving model interpretability, reducing overfitting, and enhancing generalization to unseen data.Feature selection techniques can be categorized into three types which are, Filter Methods: These methods evaluate the relevance of features based on statistical measures (e.g., correlation, mutual information) and select features independently of the model, Wrapper Methods: These methods involve evaluating different subsets of features using a specific machine learning model and selecting the subset that yields the best performance and Embedded Methods: These methods integrate feature selection within the model training process, where feature importance is determined during model training.

4. Model Selection & Model Training

Choose appropriate machine learning models for the task. Since this is a classification problem, we can consider models such as Logistic Regression, Random forests, Support Vector Machines and K Nearest Neighbor. Split the dataset into training and testing sets to evaluate the models.

The tools, libraries and frameworks used are:
- NumPy, Pandas for Data Preprocessing
- Matplotlib/ Seaborn for Visualization
- Correlation based Feature Selection
- PyTorch/ TensorFlow as framework if needed
- Grid search CV – Cross Validation
- Scikit Learn for model building and evaluation

For the model training GridSearchCV has been used to train all the models for hyper parameter tuning.

4.1. GridSearchCV for Hyperparameter Tuning:

GridSearchCV (Grid Search Cross-Validation) is a technique used for hyperparameter tuning in machine learning models. It systematically searches through a predefined grid of hyperparameters, evaluating model performance using cross-validation on each combination of hyperparameters. This helps in finding the optimal set of hyperparameters that yields the best performance for the given dataset.GridSearchCV helps in automating the process of hyperparameter tuning, saving time and effort compared to manual tuning. It also reduces the risk of overfitting by using cross-validation to estimate the model's performance on unseen data.

Here's an overview of how GridSearchCV works:

- Define the Model and Hyperparameter Grid:

Specify the machine learning model to be tuned (e.g., Support Vector Machine, Random Forest, etc.).

- Define a grid of hyperparameters to be tuned. Each hyperparameter can have multiple values to be tried.
- Cross-Validation:

Split the dataset into multiple folds (typically k-folds). For each combination of hyperparameters in the grid: Train the model on k-1 folds, Validate the model on the remaining fold, Calculate the evaluation metric (e.g., accuracy, F1-score) on the validation fold.

- Select the Best Hyperparameters:

Determine the combination of hyperparameters that maximizes the performance metric averaged across all folds.

- Train Final Model:

Train the model using the entire dataset with the best hyperparameters obtained from the grid search.

- Evaluate Model Performance:

Optionally, evaluate the final model on a separate test dataset to estimate its generalization performance.

5. Models Used

5.1. Logistic Regression:

Logistic Regression is a model for classification, particularly suitable for problems where the target variable is binary. It models the probability of a discrete outcome given an input variable. In our context, it predicts the probability of the price category based on various features. During grid search, we explore hype parameter C, which controls the regularization strength, balancing between fitting the training data and preventing overfitting.

Some common steps in using logistic regression with the Diabetic Retinopathy Debrecen Dataset could include:
1. Data Preprocessing: This involves cleaning the dataset, handling missing values, and possibly scaling or normalizing the features to ensure that they are on a similar scale.
2. Feature Selection: Identifying the most relevant features that are likely to have a significant impact on predicting diabetic retinopathy. This can be done using techniques such as correlation analysis, feature importance, or domain knowledge.
3. Model Training: Splitting the dataset into training and testing sets, then training the logistic regression model on the training data. During training, the model adjusts its parameters to minimize the error in predicting the outcome variable (presence or absence of diabetic retinopathy).
4.Model Evaluation: Evaluating the performance of the trained logistic regression model on the testing data to assess its accuracy, precision, recall, F1-score, and other relevant metrics. This helps determine how well the model generalizes to new, unseen data.

5.Model Interpretation: Analyzing the coefficients of the logistic regression model to understand the relationship between the predictor variables and the likelihood of diabetic retinopathy. This can provide insights into which features are most strongly associated with the condition.

This code first loads the dataset from the given URL, performs feature engineering (you can add additional feature engineering steps as needed), splits the data into training and testing sets, scales the features, and then uses GridSearchCV for hyperparameter tuning of the logistic regression model. Finally, it evaluates the model on the test set and prints out the best parameters and accuracy achieved.

5.2. K Nearest Neighbour (KNN)

K-Nearest Neighbors (KNN) is a supervised machine learning algorithm used for classification and regression tasks. It's a non-parametric and lazy learning algorithm, meaning it doesn't make explicit assumptions about the underlying data distribution and delays learning until it needs to make predictions. For classification, when presented with a new, unseen datapoint, KNN identifies its K nearest neighbors in the training dataset based on a chosen distance metric. Implementing the K-Nearest Neighbors (KNN) algorithm for diabetic retinopathy detection using the Debrecen dataset involves the following steps:

1. Data Preprocessing:
Load the Debrecen dataset into your programming environment. Conduct data preprocessing tasks such as handling missing values, scaling features, and encoding categorical variables if necessary.
2. Feature Selection/Extraction (Optional):
Analyze the features in the dataset and perform feature selection or extraction techniques to reduce dimensionality and improve model performance.
3. Split the Dataset:
Split the dataset into training and testing sets to evaluate the performance of the KNN algorithm.
4. KNN Model Training:
Initialize a KNN classifier with appropriate parameters such as the number of neighbors (K). Train the KNN model on the training dataset.
5. Model Evaluation:
Evaluate the trained KNN model on the testing dataset using appropriate evaluation metrics such as accuracy, precision, recall, and F1-score. Experiment with different values of K and assess their impact on model performance.
6. Interpretation and Fine-tuning:
Analyze the results and interpret the predictions made by the KNN model. Fine-tune the model parameters if necessary to improve performance.

5.3. Support Vector Machine (SVM):

Support Vector Machine (SVM) is a powerful supervised machine learning algorithm commonly used for classification tasks, including diabetic retinopathy detection. Support Vector Machines (SVM) are robust algorithms for classification tasks. SVMaims to find the hyperplane that best separates classes in a high dimensional space. This supervised learning method is effective in various domains. Our grid search covers hyperparameters such as the kernel type and regularization parameter, crucial for defining the SVM's decision boundary. Here's a brief overview of implementing SVM for the diabetic retinopathy detection task using the Debrecen dataset:

1. Data Preprocessing:
Load the Debrecen dataset into your preferred programming environment (e.g.,Python). Perform exploratory data analysis (EDA) to understand the structure and distributions of the dataset. Preprocess the data by scaling features if necessary, and encoding categorical variables.
2. Feature Selection/Extraction:
If needed, perform feature selection or extraction techniques to reduce the dimensionality of the dataset and improve model performance.
3. Split the Dataset:
Split the dataset into training and testing sets to evaluate the performance of the SVM model.
4. SVM Model Training:
Choose an appropriate SVM kernel (e.g., linear, polynomial, radial basis function (RBF)). Train the SVM model on the training dataset using the selected kernel. Tune hyperparameters (e.g., C, gamma) using techniques like GridSearchCV to optimize model performance.
5. Model Evaluation:
Evaluate the trained SVM model on the testing dataset using appropriate evaluation metrics (e.g., accuracy, precision, recall, F1-score). Visualize the performance metrics to assess the model's effectiveness in diabetic retinopathy detection.
6. Interpretation and Fine-tuning:
Analyze the results and interpret the model's predictions. Fine-tune the SVM model parameters if necessary to improve performance or address any issues identified during evaluation.

5.4. Random Forest Classifier

RandomForest is an ensemble method that combines predictions from multiple DecisionTrees.This approach mitigates overfitting and enhances predictive performance. In our context, RandomForest aggregates the output of individual decision trees, offering a robust prediction. The grid search optimizes parameters such as the number of trees, maximum depth, and other hyper parameters. Implementing a Random Forest Classifier for diabetic retinopathy detection using the Debrecen dataset involves several steps. Below is a step-by-step guide:

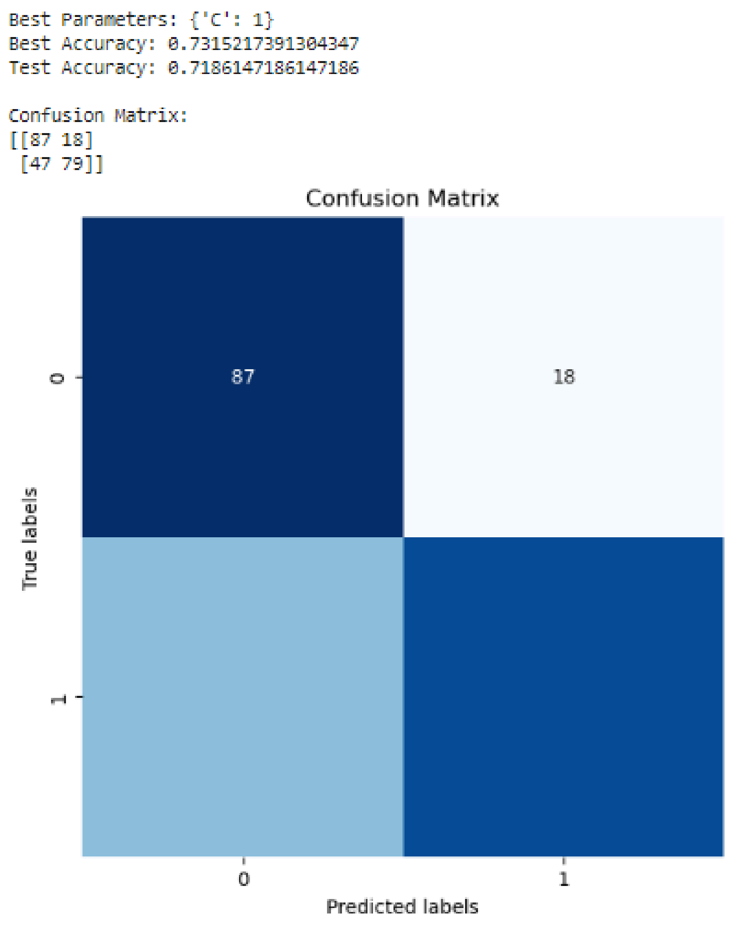
1. Data Preprocessing:
Load the Debrecen dataset into your preferred programming environment (e.g.,Python). Perform exploratory data analysis (EDA) to understand the structure and distributions of the dataset. Preprocess the data by handling missing values, scaling features if necessary, and encoding categorical variables.
2. Feature Selection/Extraction (Optional):
If needed, perform feature selection or extraction techniques to reduce the dimensionality of the dataset and improve model performance.
3. Split the Dataset:
Split the dataset into training and testing sets to evaluate the performance of the Random Forest Classifier.

4. Random Forest Model Training:
Initialize a Random Forest Classifier model with appropriate hyperparameters. Train the Random Forest model on the training dataset. Tune hyperparameters (e.g., number of trees, maximum depth, minimum samples per leaf) using techniques like GridSearchCV to optimize model performance.
5. Model Evaluation:
Evaluate the trained Random Forest model on the testing dataset using appropriate evaluation metrics (e.g., accuracy, precision, recall, F1-score). Visualize the performance metrics to assess the model's effectiveness in diabetic retinopathy detection.
6. Interpretation and Fine-tuning:
Analyze the results and interpret the model's predictions. Fine-tune the Random Forest model parameters if necessary to improve performance or address any issues identified during evaluation.

6. Model Evaluation and Validation (Results)

6.1. Logistic Regression Results and Validation

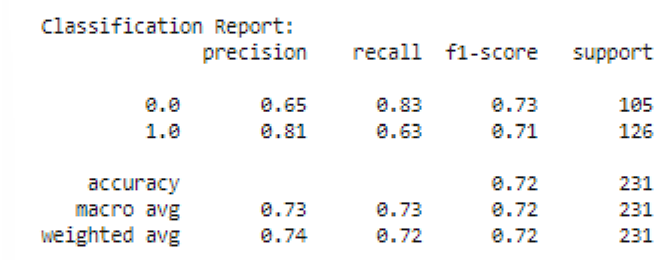
Logistic regression has given a **best accuracy of 73.1% and a test accuracy of 71.8%**. The confusion matrix is also depicted in the below Figure 7.



```
Best Parameters: {'C': 1}
Best Accuracy: 0.7315217391304347
Test Accuracy: 0.7186147186147186

Confusion Matrix:
[[87 18]
 [47 79]]
```

Fig 7: Confusion Matrix for Logistic Regression.

A classification report was also made to analyze the Precision, recall, f1-score and support as shown in Figure 8.

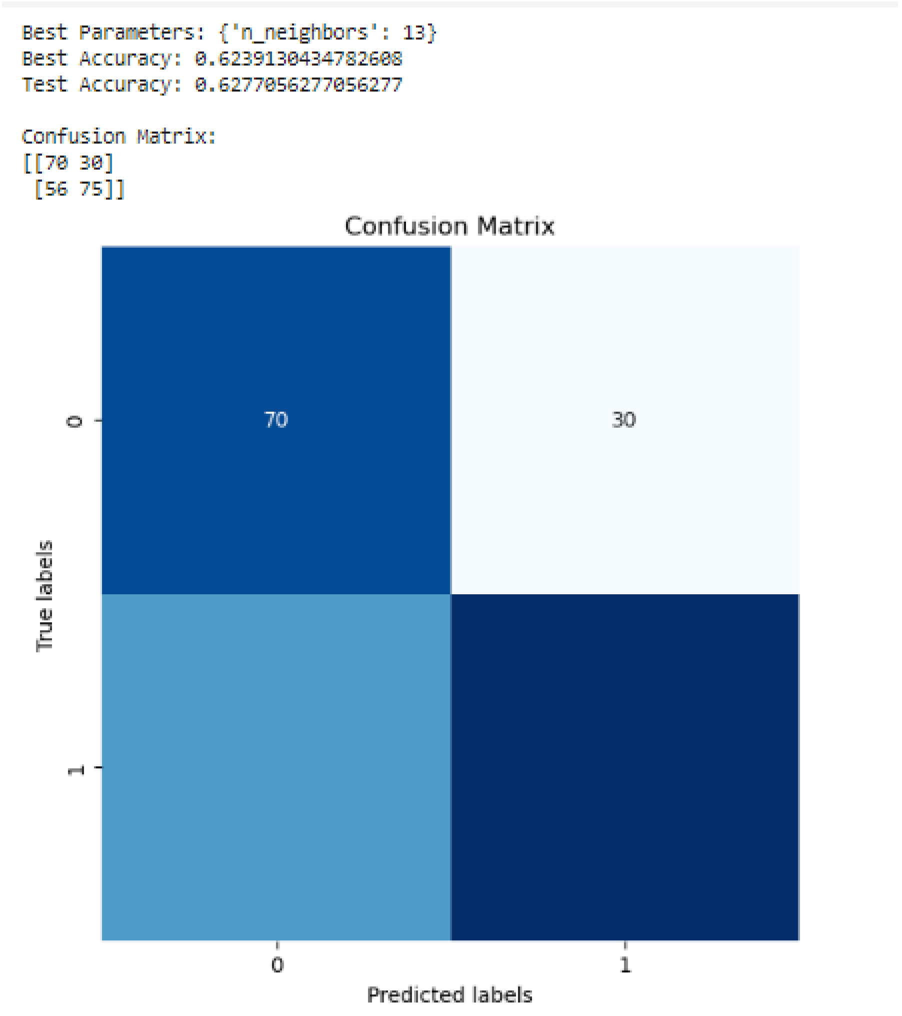


```
Classification Report:
              precision    recall  f1-score   support

         0.0       0.65      0.83      0.73       105
         1.0       0.81      0.63      0.71       126

    accuracy                           0.72       231
   macro avg       0.73      0.73      0.72       231
weighted avg       0.74      0.72      0.72       231
```

Fig 8: Classification Report for Logistic Regression

6.2. K Nearest Neighbour Results and Validation

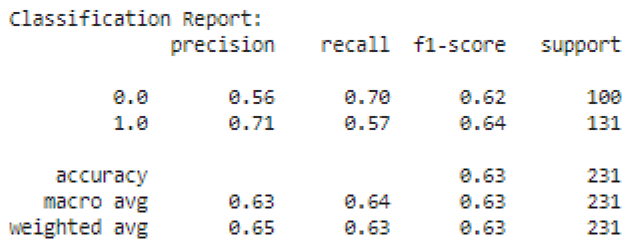
Logistic regression has given a **best accuracy of 62.3% and a test accuracy of 62.7%**. The confusion matrix is also depicted in the below Figure 9.



```
Best Parameters: {'n_neighbors': 13}
Best Accuracy: 0.6239130434782608
Test Accuracy: 0.6277056277056277

Confusion Matrix:
[[70 30]
 [56 75]]
```

Fig 9: Confusion Matrix for KNN.

A classification report was also made to analyze the Precision, recall, f1-score and support as shown in Figure 10.



```
Classification Report:
              precision    recall  f1-score   support

         0.0       0.56      0.70      0.62       100
         1.0       0.71      0.57      0.64       131

    accuracy                           0.63       231
   macro avg       0.63      0.64      0.63       231
weighted avg       0.65      0.63      0.63       231
```

Fig 10: Classification Report for KNN

6.3. Support Vector Machine Results and Validation

Logistic regression has given a **best accuracy of 72.9% and a test accuracy of 75.3%**. The confusion matrix is also depicted in the below Figure 11.

```
Best Parameters: {'C': 100, 'gamma': 0.1, 'kernel': 'rbf'}
Best Accuracy: 0.7293478260869566
Test Accuracy: 0.7532467532467533

Confusion Matrix:
[[86 17]
 [40 88]]
```

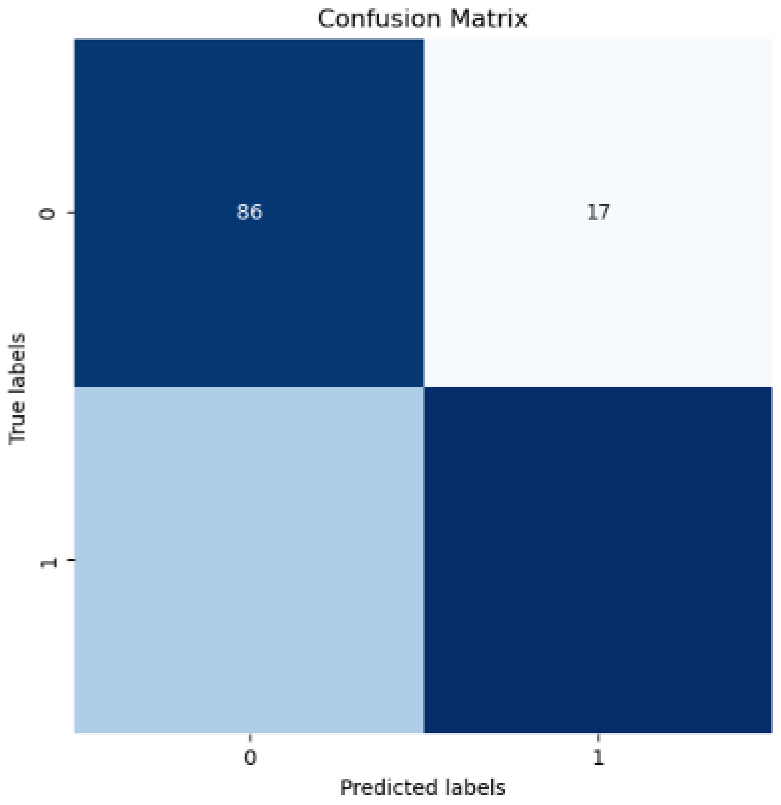
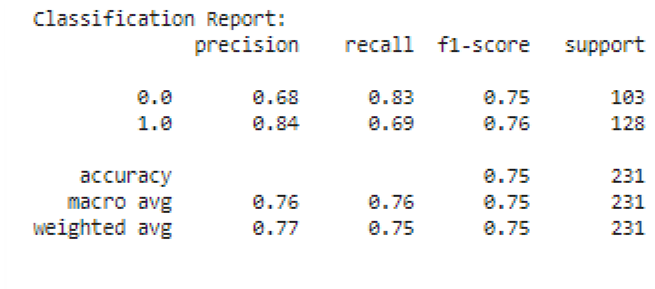


Fig 11: Confusion Matrix for SVM.

A classification report was also made to analyze the Precision, recall, f1-score and support as shown in Figure 12.



```
Classification Report:
              precision    recall  f1-score   support

         0.0       0.68      0.83      0.75       103
         1.0       0.84      0.69      0.76       128

    accuracy                           0.75       231
   macro avg       0.76      0.76      0.75       231
weighted avg       0.77      0.75      0.75       231
```

Fig 12: Classification Report for SVM

6.4. Random Forest Classifier Results and Validation

Logistic regression has given a **best accuracy of 67.7% and a test accuracy of 64.9%**. The confusion matrix is also depicted in the below Figure 13.

```
Best Parameters: {'bootstrap': False, 'max_depth': 20, 'min_samples_leaf': 2,
Best Accuracy: 0.6771739130434783
Test Accuracy: 0.6493506493506493

Confusion Matrix:
[[72 43]
 [38 78]]
```

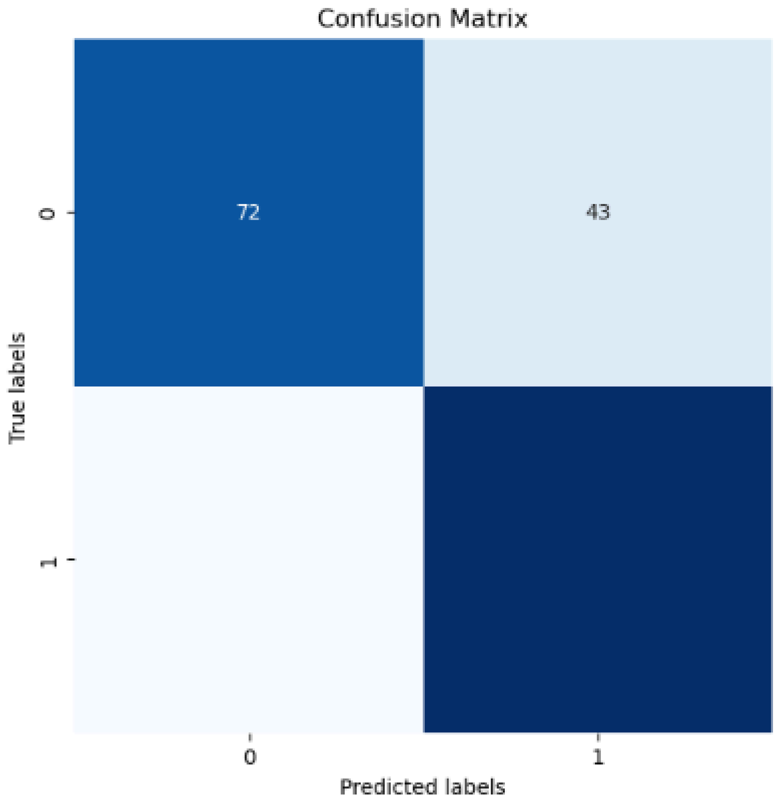
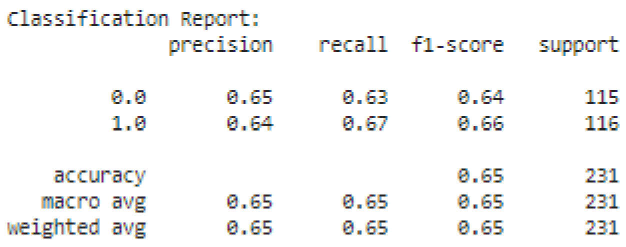


Fig 13: Confusion Matrix for RF.

A classification report was also made to analyze the Precision, recall, f1-score and support as shown in Figure 14.



```
Classification Report:
              precision    recall  f1-score   support

         0.0       0.65      0.63      0.64       115
         1.0       0.64      0.67      0.66       116

    accuracy                           0.65       231
   macro avg       0.65      0.65      0.65       231
weighted avg       0.65      0.65      0.65       231
```

Fig 14: Classification Report for RF

7. Comparison and Discussion

The comparison and discussion section aims to provide a comprehensive analysis of the model performances,drawing insights from the results obtained from both all the methods.The models were evaluated based on key metrics, including accuracy, precision, recall, and F1-score. A bar plot is created for better visualization as shown in Figure 15.

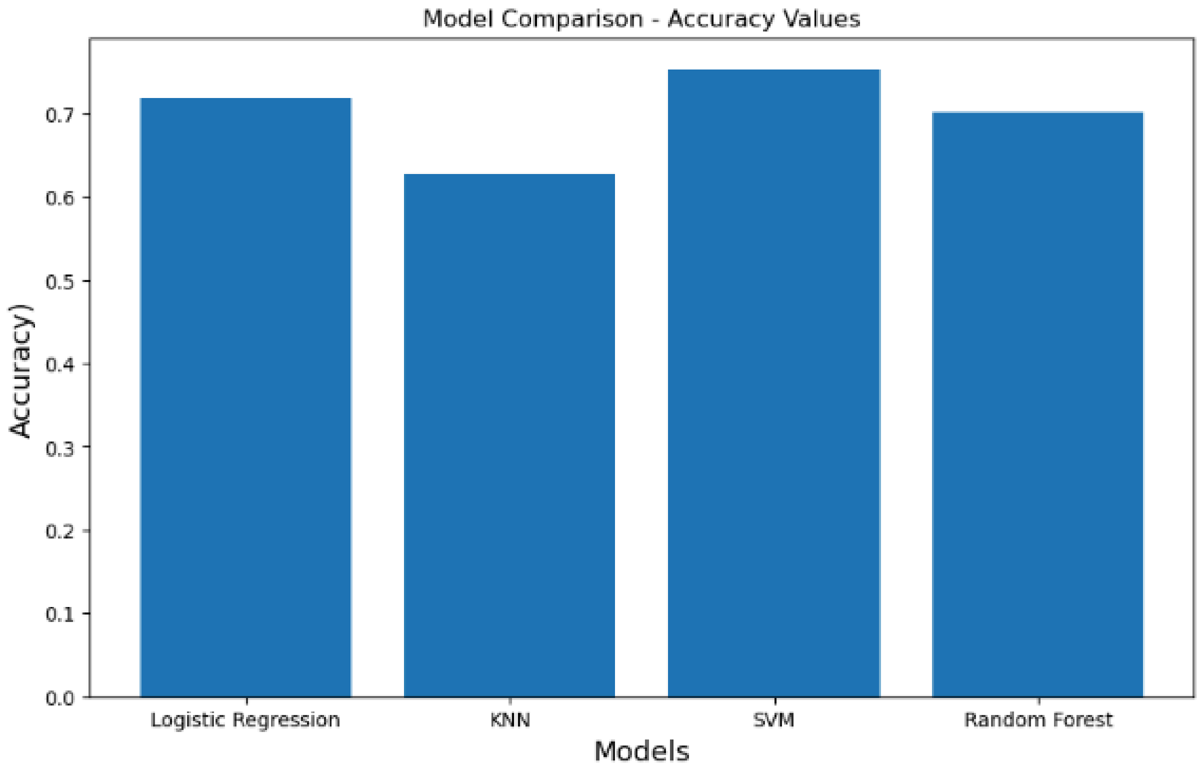


Fig 15: Model Comparison Plot

7.1. Performance Comparison

Analyzing the model performances reveals valuable insights into their strengths and weaknesses:
Accuracy: The accuracy plot illustrates the overall correctness of predictions across models with an average accuracy of around 70%. Notably, the highest accuracy was 75.3% and was from the SVM model.
Precision: The Precision plot illustrates the overall correctness of positive predictions over total predicted positive values across models with an average precision of around 76%. Notably, the highest accuracy was 84% and was from the SVM model.
Recall: The Recall plot illustrates the overall correctness of positive predictions over total positive values across models with an average recall of around 74%. Notably, the highest accuracy was 83% and was from the SVM and Logistic Regression model.
F1-Score: F1score,considering both precision and recall,provides a balanced assessment of model performance.This is not an important metric to look at especially since our dataset is balanced.The model with the highest f1-score was SVM at 76%.

Overall, SVM is the best model.

8. Future Work

In future work, an in-depth exploration into the comparative analysis between Logistic Regression, Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Random Forest for diabetic retinopathy detection could focus on several key areas. Further investigation into ensemble methods like bagging, boosting, and stacking could elucidate their potential in amalgamating the predictive power of diverse classifiers, potentially enhancing classification accuracy and robustness. Additionally, the application of transfer learning techniques, especially in CNN models, could leverage pre-trained networks to extract relevant features from retinal images, potentially boosting performance. Advanced feature engineering techniques tailored specifically for diabetic retinopathy, integration of clinical data with retinal images, and the development of explainability methods for deep learning models could offer deeper insights into disease severity and progression. Moreover, research on robustness against adversarial attacks, clinical validation studies, and seamless deployment into clinical workflows would be paramount for translating these models into practical tools for early diagnosis and management of diabetic retinopathy.

9. Conclusion

After conducting a thorough comparative analysis of five distinct machine learning and deep learning models— Logistic Regression, Support Vector Machine (SVM), Random Forest and K-Nearest Neighbors (KNN)—for diabetic retinopathy detection using the Debrecen dataset, several critical observations have been made.
Support Vector Machines (SVMs) are often preferred over logistic regression, k-Nearest Neighbors (KNN), and Random Forests in certain contexts due to their unique characteristics and capabilities. One significant advantage of SVMs lies in their effectiveness in high-dimensional spaces. In scenarios where the number of features is large relative to the number of samples.

Another notable strength of SVMs is their robustness to outliers. While logistic regression can be heavily influenced by outliers, potentially skewing the decision boundary, SVMs aim to maximize the margin between classes, thus reducing the impact of outliers. This property makes SVMs particularly useful in datasets where outliers are present or when the data distribution is not well-behaved.

Moreover, SVMs can effectively model complex, non-linear decision boundaries through the use of kernel functions. This flexibility allows them to capture intricate relationships in the data that might be challenging for logistic regression to represent directly or may require significant ensemble techniques in Random Forests. By employing different kernel functions, such as the radial basis function (RBF) kernel, SVMs can adapt to diverse datasets and learn sophisticated decision boundaries.

SVMs inherently incorporate regularization parameters to prevent overfitting, providing more control over model complexity compared to other algorithms like logistic regression and Random Forests. This regularization, combined with convex optimization techniques used in training SVMs, ensures convergence to the global minimum of the objective function, thereby enhancing generalization performance on unseen data.

Overall, while SVMs offer several advantages over logistic regression, KNN, and Random Forests in certain scenarios, the choice of algorithm ultimately depends on various factors, including dataset characteristics, computational resources, and interpretability requirements.

10. Acknowledgement

We extend our thanks to Dr. Loganantha rajan and TA, Phani Sai Ram Popuri, for their guidance and support during this project.Their expertise and insightful feedback greatly contributed to its success. We appreciate their dedication in helping us navigate complexities and shaping our approach.

11. Author Contributions

Below is a list of Authors and their contributions.
Praneeth Puppala - Software, Results, Data Curation
Tanmai Veerapaneni - Visualizations, Model Building, Writing & Reviewing, Editing
Saran Teja Mallela - Results, Model Building and Evaluation
Eeraboina Keerthi Yadav - Writing Final Draft, formalizing the comparison metrics, Validation
Kunduru Neha - Visualizations, Model Selection, Writing final Draft

References

[1] Abràmoff, M. D., Lou, Y., Erginay, A., Clarida, W., Amelon, R., Folk, J. C., & Niemeijer, M. (2016). Improved automated detection of diabetic retinopathy on a publicly available dataset through integration of deep learning. Investigative Ophthalmology & Visual Science, 57(13), 5200-5206.
[2] Bellemo, V., Lim, Z. W., Lim, G., Nguyen, Q. D., Xie, Y., Yip, M. Y., ... & Kawasaki, R. (2019). Artificial intelligence using deep learning to screen for referable and vision-threatening diabetic retinopathy in Africa: A clinical validation study. The Lancet Digital Health, 1(1), e35-e44.
[3] Gargeya, R., & Leng, T. (2017). Automated identification of diabetic retinopathy using deep learning. Ophthalmology, 124(7), 962-969.
[4] Gulshan, V., Peng, L., Coram, M., Stumpe, M. C., Wu, D., Narayanaswamy, A., ... & Webster, D. R. (2016). Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. JAMA, 316(22), 2402-2410.
[5] Kauppi, T., Kalesnykiene, V., Kamarainen, J. K., Lensu, L., Sorri, I., Raninen, A., ... & Uusitalo, H. (2007). DIARETDB1 diabetic retinopathy database and evaluation protocol. In British Machine Vision Conference (Vol. 2, No. CONF, pp. 539-539).
[6] Rajalakshmi, R., Subashini, R., Anjana, R. M., Mohan, V., & Deepa, M. (2018). Evaluation of retinal nerve fiber layer thickness measurements using optical coherence tomography in diabetic retinopathy. Indian Journal of Ophthalmology, 66(6), 815.
[7] Ting, D. S., Cheung, C. Y., Lim, G., Tan, G. S., Quang, N. D., Gan, A., ... & Wong, T. Y. (2017). Development and validation of a deep learning system for diabetic retinopathy and related eye diseases using retinal images from multiethnic populations with diabetes. JAMA, 318(22), 2211-2223.